Cocoa: What's New

Session 301

Cocoa: What’s New

Ali Ozer
Manager, Cocoa Frameworks Group

Today’s Topics

- Foundation Changes
- AppKit Changes

Foundation Changes

- Managed preferences
- NSFileManager changes
- NSNetServices
- New keyed archiving mechanism
- New binary property list format
- And more . . .

AppKit Changes

- 10.1 Changes
    - NSDocument
    - Text system
    - NSView
    - NSApplication
    - Keyboard UI

AppKit Changes

- Jaguar Changes
  - Text
  - Localized file system view
  - NSImage
  - NSWindow
  - Accessibility
  - And more . . .

Foundation Changes

Chris Parker
Foundation Engineer
Cocoa Frameworks Group

Foundation Changes

- Managed preferences
- NSFileManager changes
- NSURLHandle
- NSNetServices

Managed Preferences

- Lab environments or single machines in homes
- A “forced” domain is created at the beginning of the preference search path
- Users should not be able to change forced settings
- Applications should disable controls where the user cannot change the value of a preference key

Managed Preferences

NSUserDefaults changes

- Forced values will be returned when using objectForKey:
- Applications may need to determine if an object for a given key can be changed:

```
-(BOOL)objectIsForcedForKey: (NSString *)key;
-(BOOL)objectIsForcedForKey: (NSString *)key
        inDomain: (NSString *)domain;
```

NSFileManager

- Support for file flags
  NSFileImmutable, NSFileAppendOnly
- Creation date and HFS Codes
  NSFileCreationDate, NSFileHFSCreatorCode, NSFileHFSTypeCode
- Account information
  NSFileOwnerAccountID, NSFileGroupOwnerAccountID
- Resource fork and catalog info preserved
- Intermediate directory creation

NSURLHandle

- FTP support added
- Many bugs shaken out
- Asynchronous hostname lookup
- Integration with System Configuration planned

IPv6 in Jaguar

- Jaguar now has IPv6 support in the OS
- NSHost now returns IPv6 addresses with IPv4 addresses
- NSSocketPort now allows IPv6, IPv4 addresses in addition to host name for host arguments
- NSSocketPort prefers IPv4 addresses for compatibility
- NSURL (and CFURL) not IPv6 savvy yet

NSNetServices

- Exposes lower level OS features at Foundation level
- Dynamic discovery of TCP/IP-based services
- No prior knowledge of addresses or hosts
- Services are advertised
- Services are discovered on the network
- Discovered services are resolved for connection

NSNetService

Publishing a network service

- Services are published in a domain (local.arpa.)
- Services offered also have a name, type, and port
- Delegate object listens for events

NSNetService

Publishing a network service

- Publishing the service:

```
NSNetService *service = [[NSNetService alloc]
                          initWithDomain: @“local.arpa.”
                                    type: @“_wwdc._tcp.”
                                    name: @“Nifty Demo Server”
                                     port: 4989];
[service setDelegate: someOtherObject];
[service publish];
```

- Delegate implements:

```
netServiceWillPublish:
netService:didNotPublish:
```

NSNetServiceBrowser

Discovering domains and services

- Domains and services are dynamic
- Domains may come and go on the network
- Services appear and disappear
- An NSNetServiceBrowser informs its delegate of events as they happen

NSNetServiceBrowser

Discovering domains

- Setup:

```
NSNetServiceBrowser *domainBrowser =
                            [[NSNetServiceBrowser alloc] init];
[domainBrowser setDelegate:someOtherObject];
[domainBrowser searchForAllDomains];
```

- Delegate implements (among other things):

```
netServiceBrowser:didFindDomain:moreComing:
netServiceBrowser:didRemoveDomain:moreComing:
```

NSNetServiceBrowser

Discovering services

- Setup:

```
NSNetServiceBrowser *serviceBrowser =
                          [[NSNetServiceBrowser alloc] init];
[serviceBrowser setDelegate:someOtherObject];
[serviceBrowser
                  searchForServicesOfType: @“_wwdc._tcp.”
                                 inDomain: @“local.arpa.”];
```

- Delegate implements:

```
netServiceBrowser:didFindService:moreComing:
netServicesBrowser:didRemoveService:moreComing:
```

NSNetServiceBrowser

Resolving a discovered service

```
- (void) netServiceBrowser: (NSNetServiceBrowser *)br
               didFindService: (NSNetService *)service
                  moreComing: (BOOL)moreComing
{
        [aNetService retain];
        [aNetService setDelegate:self];
        [aNetService resolve];
}
```

NSNetServices

Additional Notes

- API is asynchronous (requires a run loop)
- NSNetService objects created for resolving cannot be used to publish
- NSNetServiceBrowser objects can only search for one thing at a time

Foundation Changes

Chris Kane
Foundation Engineer
Cocoa Frameworks Group

More Foundation Changes

- New keyed archiving mechanism
- New binary property list format and APIs
- Version checking
- More . . .

NSArchiver Issues

- Values must be unarchived in the same order as they were archived
- All archived values must be unarchived
- Cannot probe the archive for values which might not be there
- Reading new archives on older systems is cumbersome

Keyed Archiving

- Saved values are given string names
- Values can be unarchived in any desired order
- Can choose to unarchive only the values you want
- Can request values which may not be present
    - Decode methods return default values (nil, 0)

Keyed Archiving

- Simplifies backward and forward compatibility
- Type checking is still done to match encodes and decodes
- Some coercions are allowed
    - float <–> double
    - 32-bit int <–> 64-bit int

Keyed Archiving API

- New classes: NSKeyedArchiver, NSKeyedUnarchiver
- NSCoding protocol remains the same
- New NSCoder methods which take string “key” to identify the value

```
- (void)encodeObject:(id)obj forKey:(NSString *)key;
- (void)encodeBool:(BOOL)b forKey:(NSString *)key;
- (id)decodeObjectForKey:(NSString *)key;
- (BOOL)decodeBoolForKey:(NSString *)key;
```

Keyed Archiving API

- Not all NSCoder subclasses will “do” keyed archiving
- New method to test a coder for keyed-coding capabilities

```
- (BOOL)allowsKeyedCoding
```

- Currently only NSKeyedArchiver and NSKeyedUnarchiver allow keyed archiving

Keyed Archiving Example

```
// code in encodeWithCoder:
[archiver encodeObject:obj1 forKey:@“Obj1”];
[archiver encodeInt:flags forKey:@“Flags”];
[archiver encodeObject:obj2 forKey:@“Obj2”];

// code in initWithCoder:
obj2 = [unarchiver decodeObjectForKey:@“Obj2”];
shape = [unarchiver decodeIntForKey:@“Shape”];
flags = [unarchiver decodeFloatForKey:@“Flags”];
```

Keyed Archiving Example

```
// code in encodeWithCoder:
[archiver encodeObject:obj1 forKey:@“Obj1”];
[archiver encodeInt:flags forKey:@“Flags”];
[archiver encodeObject:obj2 forKey:@“Obj2”];

// code in initWithCoder:
obj2 = [unarchiver decodeObjectForKey:@“Obj2”];
shape = [unarchiver decodeIntForKey:@“Shape”];
flags = [unarchiver decodeFloatForKey:@“Flags”];
```

Keyed Archiving Example

```
// code in encodeWithCoder:
[archiver encodeObject:obj1 forKey:@“Obj1”];
[archiver encodeInt:flags forKey:@“Flags”];
[archiver encodeObject:obj2 forKey:@“Obj2”];

// code in initWithCoder:
obj2 = [unarchiver decodeObjectForKey:@“Obj2”];
shape = [unarchiver decodeIntForKey:@“Shape”];
flags = [unarchiver decodeFloatForKey:@“Flags”];
```

Keyed Archiving Example

```
// code in encodeWithCoder:
[archiver encodeObject:obj1 forKey:@“Obj1”];
[archiver encodeInt:flags forKey:@“Flags”];
[archiver encodeObject:obj2 forKey:@“Obj2”];

// code in initWithCoder:
obj2 = [unarchiver decodeObjectForKey:@“Obj2”];
shape = [unarchiver decodeIntForKey:@“Shape”];
flags = [unarchiver decodeFloatForKey:@“Flags”];
```

Keyed Archiving Example

```
// code in encodeWithCoder:
[archiver encodeObject:obj1 forKey:@“Obj1”];
[archiver encodeInt:flags forKey:@“Flags”];
[archiver encodeObject:obj2 forKey:@“Obj2”];

// code in initWithCoder:
obj2 = [unarchiver decodeObjectForKey:@“Obj2”];
shape = [unarchiver decodeIntForKey:@“Shape”];
flags = [unarchiver decodeFloatForKey:@“Flags”];
```

Keyed Archiving Example

```
// code in encodeWithCoder:
[archiver encodeObject:obj1 forKey:@“Obj1”];
[archiver encodeInt:flags forKey:@“Flags”];
[archiver encodeObject:obj2 forKey:@“Obj2”];

// code in initWithCoder:
obj2 = [unarchiver decodeObjectForKey:@“Obj2”];
shape = [unarchiver decodeIntForKey:@“Shape”];
flags = [unarchiver decodeFloatForKey:@“Flags”];
                    // last line is an error //
```

Keyed Archiving Example

```
// code in encodeWithCoder:
[archiver encodeObject:obj1 forKey:@"Obj1"];
[archiver encodeInt:flags forKey:@"Flags"];
[archiver encodeObject:obj2 forKey:@"Obj2"];

// code in initWithCoder:
obj2 = [unarchiver decodeObjectForKey:@"Obj2"];
shape = [unarchiver decodeIntForKey:@"Shape"];
flags = [unarchiver decodeFloatForKey:@"Flags"];
```

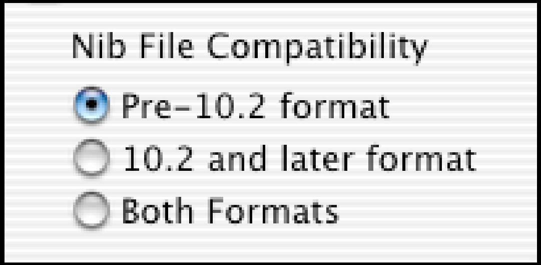
Keyed Archiving

- Can encode as many values as desired at the top level of the archive
- Should use unique prefix on string key names
- Can use inherited non-keyed coding methods

  `encodeValueOfObjCType:at:`

Keyed Archiving Output

- Output of keyed archiving
  - XML format property list
  - New binary format property list
- IB allows you to save a .nib document using NSKeyedArchiver, NSArchiver, or both



Nib File Compatibility

[✓ Pre-10.2 format]

10.2 and later format

Both Formats

Keyed Archiving

- See Foundation release notes for more information
    - Converting existing classes
    - How to plan for and do compatibility

New Property List Formats

- New binary property list format
  - More compact
  - Faster, especially for big property lists
  - Additional capabilities after Jaguar
- XML property list format has been set to 1.0
  - No changes from “0.9”

Changes to Property List API

- New class: NSPropertyListSerialization
- NSSerializer, NSDeserializer are deprecated
    - Also do not support the new property list types

Using New API and Features

- First use ObjC runtime checks
  - Check for classes with NSClassFromString()
  - Check for methods with -respondsToSelector:
- Current version global variables

```
double NSFoundationVersionNumber
double NSAppKitVersionNumber
```

Using New API and Features

- Macros recording older versions

```
#define NSFoundationVersionNumber10_1  425.0
```

- Sometimes version macros in the release notes

```
#define NSAppKitVersionNumberWithZZZ  635.0
```

Avoiding Using New API

- <AvailabilityMacros.h>
- Set MAC_OS_X_VERSION_MAX_ALLOWED
    - MAC_OS_X_VERSION_10_1
- Tips
    - Do not subclass new classes
    - Do not reference new global variables
- See the release notes on the system

Unicode 3.2

- Foundation updated to support Unicode 3.2
- Unicode characters range from 0x0 to 0x10FFFF
- NSString model based on 16-bit characters
- Characters higher than 0xFFFF represented by Unicode surrogate pairs
- Code which examines characters should use

  ```
  - (NSRange) rangeOfComposedCharacterSequenceAtIndex:
  ```

New NSString API

- Methods to trim and pad ends of strings
- Precompose, decompose strings per Unicode 3.2
- New method to efficiently replace all occurrences of one string in another

```
- (unsigned)replaceOccurrencesOfString:(NSString *)str
                            withString:(NSString *)replacementString
                               options:(unsigned)optFlags
                                 range:(NSRange)searchRange
```

NSString API Stricter

- Various methods in NSString, NSMutableString now check arguments more rigorously for nil and out-of-bounds
    - These are programming errors—FIX
    - Compatibility for apps linked on 10.1
- UTF-8 conversions now less forgiving about invalid UTF-8 sequences

Executing on the Main Thread

- New NSObject method

```
- (void)performSelectorOnMainThread:(SEL)aSel
                  withArgument:(id)arg
                  waitUntilDone:(BOOL)wait
                  modes:(NSArray *)modeList
```

- Performs methods as a result of the run loop
- Ordering for a particular thread's use is mostly maintained, but no ordering amongst threads

AppKit Changes

Ali Ozer
Manager, Cocoa Frameworks Group

10.1 AppKit Changes

NSDocument

- Now has the ability to track documents
- Preserves attributes and aliases to documents
- Supports hidden extensions

Hidden Extension Support

- File extensions are a good idea for compatibility with other systems and the web
- However, they confuse some users and disgust others
- So, allow the file extension to be hidden
    - It’s still in the name
    - But not displayed

Hidden Extension Support

- When putting up a save panel, indicate you support hidden extensions

```
- (void) setCanSelectHiddenExtension: (BOOL)flag;
```

- When save panel is dismissed, ask it whether the user wanted the extension hidden or not

```
- (BOOL) isExtensionHidden;
```

- The filename from the save panel will have the extension in either case

Hidden Extension Support

- If extension is hidden, use NSFileManager to set this in the saved document's name

```
NSMutableDictionary *attrs =
                              [NSMutableDictionary dictionary];

[attrs setObject: [NSNumber numberWithBOOL:YES]
             forKey: NSFileExtensionHidden ];

[[NSFileManager defaultManager]
   changeFileAttributes: attrs atPath: documentPath];
```

Hidden Extension Support

- When displaying document names, be sure to use the display name API in NSFileManager

```
- (NSString *) displayNameAtPath: (NSString *)path
```

Hidden Extension Support

- But wait! NSDocument does all this for you
- Supporting methods are available, if needed:

```
- (BOOL) fileNameExtensionWasHiddenInLastRunSavePanel;

- (NSDictionary *) fileAttributesToWriteToFile: (NSString *)f
                                        ofType: (NSString *)type
                              saveOperation: (NSSaveOperationType)op;

- (NSString *) displayName;
```

Text System

- Supports filter services
- Knows how to speak

Filter Services in Text

- Following NSTextStorage methods cause filter services to be invoked for unknown file types:

```
- (id) initWithPath: (NSString *)path
          documentAttributes: (NSDictionary **)dict;

- (id) initWithURL: (NSURL *)url
          documentAttributes: (NSDictionary **)dict;

- (BOOL) readFromURL: (NSURL *)url
                     options: (NSDictionary *)options
      documentAttributes: (NSDictionary **)dict;
```

Filter Services in Text

- New API to query types:

```
+ (NSArray *) textUnfilteredFileTypes;
+ (NSArray *) textUnfilteredPasteboardTypes;
+ (NSArray *) textFileTypes;
+ (NSArray *) textPasteboardTypes;
```

- And a new document attribute to find out whether a filter service was used in opening a document:

```
@“Converted”
```

Speaking Text

- New first responder action methods:

```
- (void) startSpeaking: (id)sender;
- (void) stopSpeaking: (id)sender;
```

- Implemented by NSTextView
- Available to others responders to implement

NSView

- Live resizing API for performance
  - Sent once per during a live resize “session”:

```
- (void) viewWillStartLiveResize;
- (void) viewDidEndLiveResize;
```

    Call super methods from these

    Can do setNeedsDisplay:YES at end

  - Additional convenience method:

```
- (BOOL) inLiveResize;
```

NSApplication

- Ability to set contents of dock menu
  - Provide an NSMenu in a nib

    And specify the nib as AppleDockMenu in Info.plist
  - Or via a delegate method

```
- (NSMenu *) applicationDockMenu: (id)app;
```

NSApplication

- Dock notifications

```
typedef enum {
    NSCriticalRequest,
    NSInformationalRequest
} NSRequestUserAttentionType;

- (int) requestUserAttention: (NSRequestUserAttentionType)t;
- (void) cancelUserAttentionRequest: (int)request;
```

- Automatic for modal panels in inactive apps

Keyboard UI

- Came back to life 10.1
- Windows with initialFirstResponder are assumed to have valid keyboard UI loops
    - Otherwise the kit computes one for you

Keyboard UI

- API to draw focus rings

```
typedef enum {
    NSFocusRingOnly,
    NSFocusRingBelow,
    NSFocusRingAbove
} NSFocusRingPlacement;

void NSSetFocusRingStyle (NSFocusRingPlacement p);
```

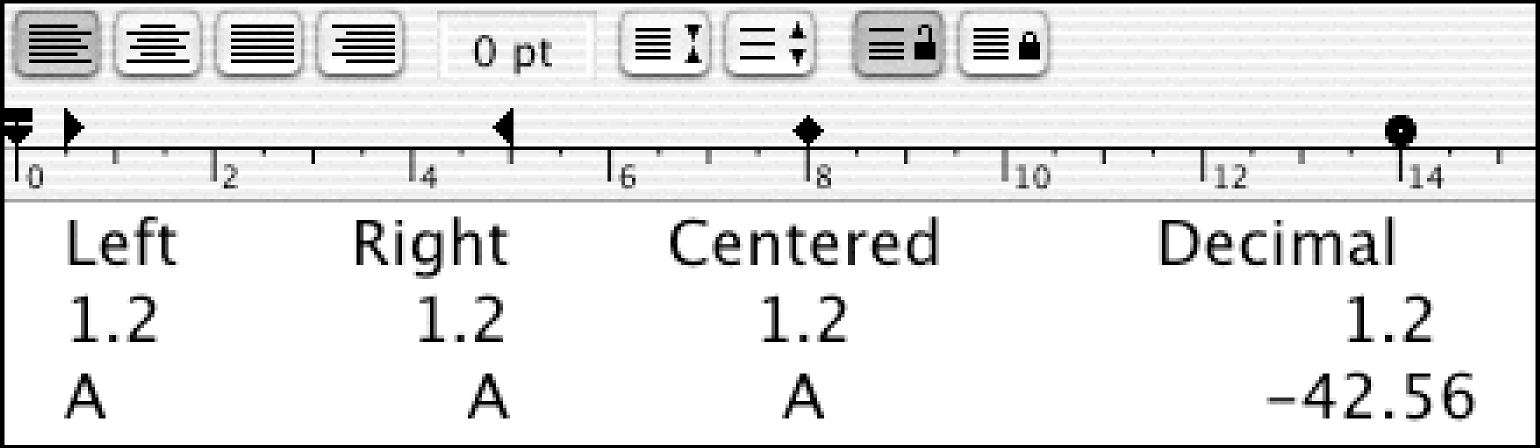
- More discussion in “Cocoa Controls and Cocoa Accessibility” talk

Jaguar AppKit Changes

Text System

- Comes with more built-in spelling checkers
    - British English, anyone? A colorful language
- Supports right, centered, and decimal tab stops
- Does bidirectional text

Tab Stops



```
typedef enum {
    NSLeftTabStopType = 0,
    NSRightTabStopType,
    NSCenterTabStopType,
    NSDecimalTabStopType
} NSTextTabType;
```

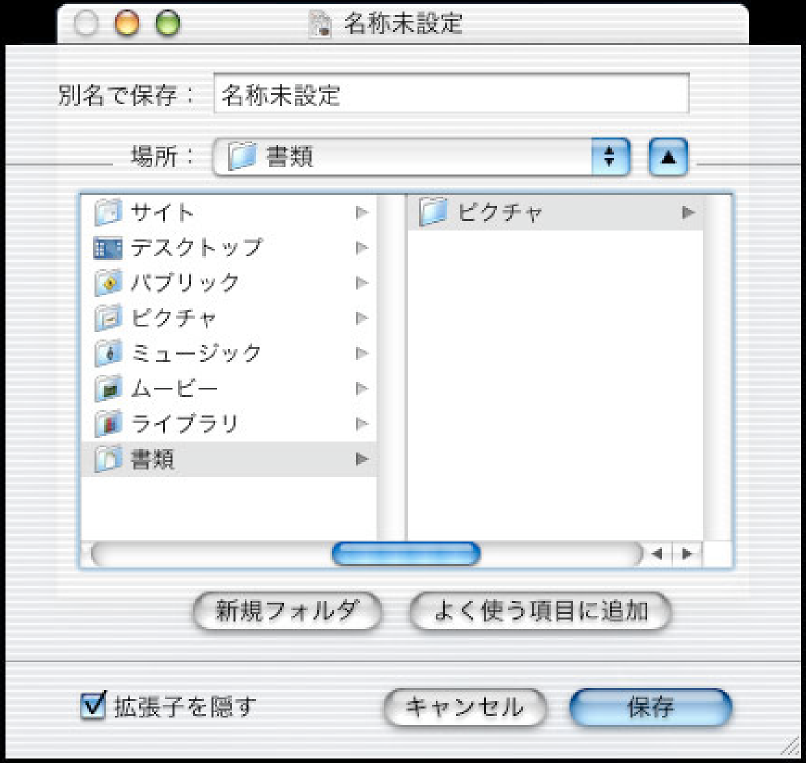
Bidirectional Text

- Needed for proper support for Hebrew, Arabic, and some other scripts
- New API
    - New NSLayoutManager glyph attribute

      NSGlyphAttributeBidiLevel
    - New NSTypesetter subclass

Localized File System View

- Present files with localized names
    - Not localizing the file system
    - But the view of the file system


別名で保存： 名称未設定
場所： 書類

Localized File System View

- When displaying document names, be sure to use the display name API in NSFileManager

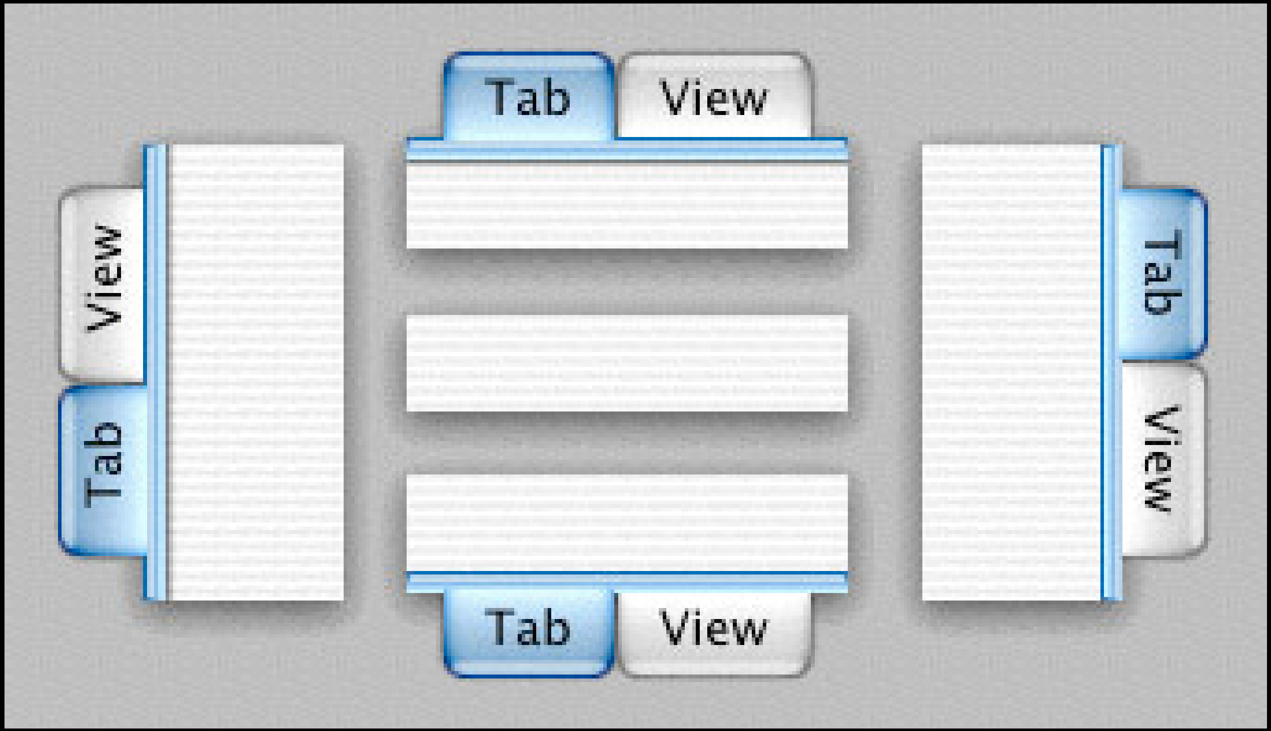
```
- (NSString *) displayNameAtPath: (NSString *)path
```

Localizing Your App Name

- Add CFBundleName to Info.plist
  - Same as the bundle’s actual name
- Add the key LSHasLocalizedDisplayName with boolean value true
- Add CFBundleName in your InfoPlist.strings

NSTabView

- Directional tabs



Directional Tabs

- No new APIs
- These existing NSTabItem APIs still work as before:

```
- (void) drawLabel: (BOOL)truncate
              inRect: (NSRect)labelRect;

- (NSSize) sizeOfLabel: (BOOL)computeMin;
```

NSImage

- Animated images
- Progressive image loading
- Caching policy

Animated Images

- Additional properties on NSImageReps created from animated images:

  NSImageFrameCount
  NSImageCurrentFrame
  NSImageCurrentFrameDuration

Progressive Image Loading

- Via NSImage delegate methods:

```
- (void) image: (NSImage*)image
            willLoadRepresentation: (NSImageRep*)r;

- (void) image: (NSImage*)image
            didLoadRepresentationHeader: (NSImageRep*)r;

- (void) image: (NSImage*)image
            didLoadPartOfRepresentation: (NSImageRep*)r
            withValidRows: (int)rows;

- (void) image: (NSImage*)image
            didLoadRepresentation: (NSImageRep*)r
            withStatus: (NSImageLoadStatus)status;
```

Caching Policy

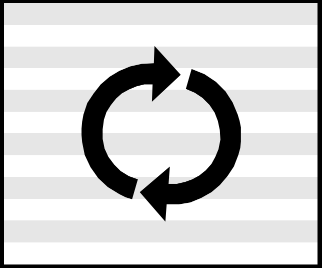
- Makes it explicit whether NSImage should cache its images (in off-screen windows)

```
typedef enum {
    NSImageCacheAlways,
    NSImageCacheBySize,
    NSImageCacheNever
} NSImageCachingMode;

- (void) setImageCacheMode: (NSImageCachingMode)m;
- (NSImageCachingMode) imageCacheMode;
```

Spinning (aka Chasing) Arrows

- NSProgressIndicator variant



```
typedef enum {
    NSProgressIndicatorBarStyle = 0,
    NSProgressIndicatorSpinningStyle
} NSProgressIndicatorStyle;

- (void) setStyle: (NSProgressIndicatorStyle) style;
- (NSProgressIndicatorStyle) style;
```

Spinning Arrows

- Additional API

```
- (void) sizeToFit;

- (BOOL) isDisplayedWhenStopped;
- (void) setDisplayedWhenStopped: (BOOL) flag;
```

- These apply to “bar” style as well as “spinning”

NSToolbar

- Now has small icon mode

```
typedef enum {
   NSToolbarSizeModeDefault,
   NSToolbarSizeModeRegular,
   NSToolbarSizeModeSmall
} NSToolbarSizeMode;

- (void) setSizeMode: (NSToolbarSizeMode)mode;
- (NSToolbarSizeMode) sizeMode;
```

NSWorkspace

- Application notifications now include the following keys for additional info:
    - NSApplicationPath, NSApplicationName

      NSStrings containing full path and name
    - NSApplicationProcessIdentifier

      NSNumbers containing process id
    - NSApplicationProcessSerialNumberHigh/Low

      NSNumbers containing high and low parts of PSN

NSWorkspace

- Also has new methods for getting information about running applications:

```
- (NSArray *) launchedApplications;
- (NSDictionary *) activeApplication;
```

- These contain the same keys and values described on the previous slide

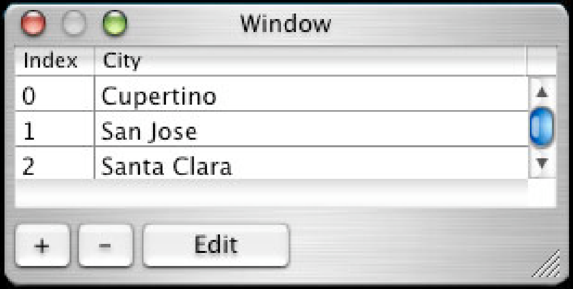
NSWindow

- New style of panel

  NSNonactivatingPanelMask

- New style of window

  NSTexturedBackgroundWindowMask



Index	City
0	Cupertino
1	San Jose
2	Santa Clara

NSWindow

- Ability to move windows from the content area

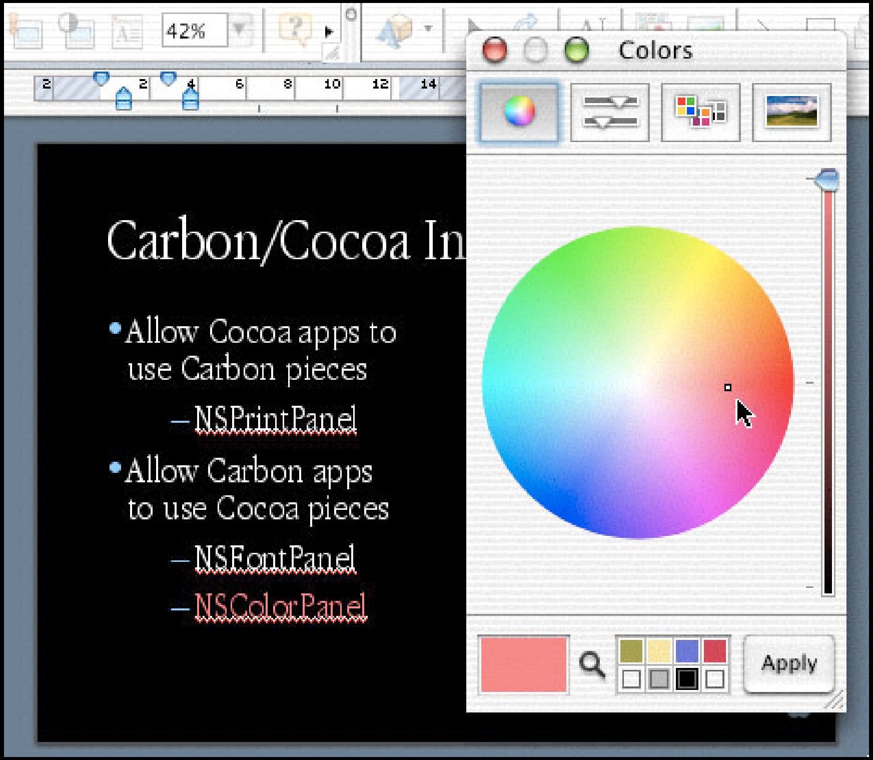
  ```
  - (void) setMovableByWindowBackground: (BOOL)flag;
  ```

- Ability to set background colors/patterns

  ```
  - (void) setBackgroundColor: (NSColor *)color;
  ```

Carbon/Cocoa Integration

- Allow Cocoa apps to use Carbon pieces
    - NSPrintPanel
- Allow Carbon apps to use Cocoa pieces
    - NSFontPanel
    - NSColorPanel



Carbon/Cocoa In

- Allow Cocoa apps to use Carbon pieces
    - NSPrintPanel
- Allow Carbon apps to use Cocoa pieces
    - NSFontPanel
    - NSColorPanel

Cocoa/Carbon Integration

- Create an NSWindow around a Carbon window:
  ```
  - (NSWindow *) initWithWindowRef: (WindowRef)ref;
  ```
- Return the Carbon window:
  ```
  - (WindowRef)windowRef;
  ```
- Handle events via CarbonEvents

Cocoa/Carbon Integration

- It’s also possible to load Cocoa bundles in Carbon apps
    - Load bundle with CFBundle
    - Initialize Cocoa with NSApplicationLoad()
- NSColorPanel
- NSFontPanel

NSPasteboard

- New pasteboard data type

  NSVCardPboardType
- And a brand new framework with Cocoa APIs for your other address-book and related needs

  AddressBook.framework
- See “Address Book Framework” talk for details

Accessibility

- Enable writing assistive applications
  - Screen readers
  - Alternate input devices
- C APIs for assistive application creators

  AXUIElementSetAttributeValue(), …
- Cocoa APIs for providing accessibility features in Cocoa applications

Accessibility

- Cocoa UI elements implement the NSAccessibility informal protocol
    - Presents the user interface as a hierarchy of “UI Elements” with attributes and actions
    - Many built-in controls implement this
    - Many subclassers also automatically benefit
- See “Cocoa Controls and Cocoa Accessibility”

Scripting

- NSAppleScript
    - Load, compile, and execute scripts
    - Get “pretty printed” scripts
- Automatic support for “Properties” property
- Better conversions
- See “Cocoa Scripting”

Release Notes

- Find Jaguar release notes on your Jaguar CD at:
  /Developer/Documentation/ReleaseNotes/
    AppKit.html
    Foundation.html

Cocoa Documentation

- Object-Oriented Programming and the Objective-C Language
- Programming Topics

Application Architecture	Memory Management
Foundation Framework	Multithreading
Loading Resources	Notifications

. . .and many more!

Documentation > Cocoa
developer.apple.com/techpubs/macosx/Cocoa/CocoaTopics.html

For More Information

- O'Reilly "Learning Cocoa" and "Building Cocoa Applications: A Step-by-Step Guide"
- Cocoa Developer Documentation
  http://developer.apple.com/techpubs/macosx/Cocoa/CocoaTopics.html
- Apple Customer Training
  http://train.apple.com/

Roadmap

302 Cocoa API Techniques: Understanding, leveraging, and extending	Hall 2 **Thurs., 9:00am**
303 Cocoa Scripting: Scripting overview and recent changes	Room A2 **Thurs., 10:30am**
304 Cocoa Controls and Accessibility: Overview of controls; new Accessibility APIs	Room A2 **Thurs., 5:00pm**
305 Cocoa Drawing: Drawing using Cocoa APIs	Hall 2 **Fri., 10:30am**
306 Cocoa Text: In-depth overview of the text system	Room J **Fri., 2:00pm**

Roadmap (Cont.)

805 Introducting CFNetwork: CF APIs for networking and services	Room C **Tue., 5:00pm**
811 Zero Configuration Networking: Support for services, dynamic configuration	Room J **Thurs., 2:00pm**
012 Address Book Framework: Overview of new Address Book APIs	Room C **Fri., 3:30pm**
FF016 Cocoa: Comments and suggestions for Cocoa	Room A1 **Fri., 5:00pm**

Who to Contact

Heather Hickman
Cocoa Technology Manager
hhickman@apple.com

Q&A

Heather Hickman
Cocoa Evangelist
hhickman@apple.com

http://developer.apple.com/wwdc2002/urls.html

WWDC 2002

WWDC2002

WWDC2002